Diving into Cloud-based File Synchronization with User Collaboration

Haiyang Wang[1], Xiaoqiang Ma[2], Feng Wang[4], Jiangchuan Liu[2,3], Bharath Kumar Bommana[1], Xin Liu[5]*
[1] University of Minnesota Duluth, USA, Email: {Haiyang, bomma008}@d.umn.edu
[2]Simon Fraser University, Burnaby, Canada, Email: {xma10, jcliu}@cs.sfu.ca
[3]South China Agricultural University, Guangzhou, China
[4]The University of Mississippi, University, USA, Email: fwang@cs.olemiss.edu
[5]Tsinghua University, China, xinliutsinghua@gmail.com

***Abstract*—In this paper, we take a close look to understand the cloud-based file synchronization and collaboration systems. Using the popular Dropbox as a case study, our measurement reveals its cascaded computation and communication operations that are far more complicated than those in conventional file hosting. We show that this serial design is necessary for the cloud deployment, which effectively avoids the possible task interference inside the computation cloud; yet it also leads to higher service variance across users. Even worse, in a collaborative file editing session, users' updates would be discarded without any warning. The drop rate is unfortunately related to the slowest collaborator, which severely hinders the system scalability and user satisfaction. We further investigate the root causes of this phenomenon as well as other performance bottlenecks and offer hints for practical improvement.**

I. INTRODUCTION

Recent years have witnessed the rising popularity of cloud-based file storage systems. Such commercial products as Dropbox [1], Gdrive [2] and Skydrive [3] not only provide reliable file hosting but also enable effective file synchronization for user collaboration. It is known that this new generation of file synchronization is greatly benefited from *cloud computing*. For example, Dropbox is using Amazon's S3 service for file storage; Gdrive and Skydrive, on the other hand, utilize the cloud platforms by Google and Microsoft, respectively. The richer services as well as abundant computation, storage, and communication resources enabled by the clouds, way beyond those by conventional servers or CDNs, no doubtable contribute to the tremendous success of these systems.

It is however known that the QoS (quality of service) of such synchronization systems is far from being satisfactory, particularly in terms of synchronization latency. With the expansion of system scale, it frequently exceeds the accepted level for practical collaboration, which has been widely complained[1]. Unfortunately, the framework design and protocol operation of the systems remain vague to the general public. Identifying the exact performance bottlenecks or enabling theoretical and practical optimization is acutely challenging and largely blinding to date.

[1]For example, in the official Dropbox forum http://forums.dropbox.com/topic.php?id=12859.
*Corresponding Author

In this paper, we reveal the cascaded stages during Dropbox's file synchronization, namely, *pre-processing*, *uploading*, *downloading*, and *post-processing*. We show that this series of computation and communication operations, which is far more complicated than those in conventional file hosting [4], is necessary for Dropbox-like services especially considering the cloud deployment. Such a design can significantly improve service reliability and avoid the possible task interference on cloud-based virtual machines (VMs); yet it also leads to higher latency and cost. In particular, the variance of latency across different users increases with larger population, and thus individual users may face severe performance degradation when the system scale grows.

Even worse, our measurement shows that, in a collaborative file editing session, Dropbox would discard users' updates without any warning. The drop rate is unfortunately related to the slowest (in terms of latency) collaborator in the system, which severely hinders the system scalability. To ensure reliability, a user may have to wait tens of seconds for each (even the smallest) update when the number of its collaborators exceeds 10. This *safe interval* ramps up quickly, as revealed by our measurement across different numbers of users under various network conditions. We further investigate the root causes of this phenomenon and other performance bottlenecks by diving into the cloud infrastructure of Dropbox-like systems and offer hints of practical improvement.

The rest of this paper is organized as follows. In Section II, we present the related works. After that, we decompose the synchronization latency of Dropbox in Section III. Section IV explores the synchronization performance of Dropbox, and Section V explores the collaboration among the users. Section VI discusses the root causes of dropping updates and offers hints for practical improvement. Section VII concludes the paper.

II. BACKGROUND AND RELATED WORKS

File storage and synchronization has long been a critical service in the IT industry. Since the first development of RAID (redundant array of independent disks) [5], many storage and synchronization technologies have been proposed for local and networked environments. In the Internet, client-server- or CDN-based protocols [4] and peer-assisted protocols [6] both have seen wide deployment; they mainly support file hosting, with no or limited user collaborations.

Cloud computing provides elastic service, powerful capacity and low cost to service providers. These salient features have opened new opportunities to support Internet applications on cloud platforms [7] [8] [9]. Wu *et al.* [10] explored the use of cloud for video-on-demand applications; Kannan *et al.* [11] examined the optimization of home clouds for mobile devices. Many studies have also addressed application designs that leverage cloud platforms [12] [13] or optimize the cloud traffic [14] [15]. A number of companies have also been enticed to deploy their own cloud-based file storage and synchronization systems [1] [2] [3] [16]. To better understand these systems, Drago *et al.* [17]analyzed the system workload of Dropbox. They found that the Dropbox performance is mainly driven by the distance between clients and storage data centers. A recent study from Li *et al.* [18] showed that a considerable portion of the cloud sync traffic is in a sense wasteful, and can be effectively avoided or significantly reduced via carefully designed data sync mechanisms.

Different from the existing studies, our work is focusing on the protocol of cloud-based synchronization and collaboration. We reveals the cascaded stages during Dropbox's file synchronization and identify a severe bottleneck of cloud-based file synchronization, shedding new light on future explorations.

III. Design Decomposition of File Synchronization

In this section, we first discuss the most widely used core functionality of Dropbox, namely file synchronization. We provide a detailed measurement, which decomposes the synchronization process and reveals the associated overhead.

We start from an experiment of two Dropbox users. We select two nodes from the planet-lab platform to deploy the Dropbox client application, one as the data source that uploads the file, and the other as the destination that needs to be synchronized. The data source is located in the University of British Columbia (UBC) and the destination is deployed in Simon Fraser University (SFU). Both nodes have similar hardware capacity with 1.7 GHz CPU, 4 GB memory and 1 Gbps Ethernet adaptor. We link these two nodes with the same Dropbox account. Once we move a file to the Dropbox folder, this file will be synchronized to another PC, and we use *synchronization latency* to represent the time span of the whole synchronization process. We are focusing on two metrics that are closely related to the synchronization latency: the CPU utilization, and the downloading/uploading rates on these two PCs.

We first synchronize a 500 MBytes file between these two PCs. In the experiment, we generate the file by writing random characters. After each synchronization, we randomly shuffle the file to avoid the cache functions of Dropbox and ensure the whole file is uploaded. Otherwise Dropbox will only update the changed part.

Figure 1 and Figure 4 present the CPU utilization and the downloading/uploading rate on these two Dropbox users, respectively. The solid lines refer to the data source, and the dotted lines refer to the destination. As we can see from Figure 1, the CPU utilization elevates sharply in the first 20 seconds after we put the file into the Dropbox folder. During this time, there is no high-speed data transmission in Figure 4. A closer look shows that this elevating CPU load corresponds to such file pre-processing as splitting the files into chunks and computing their hash values using the SHA-256 algorithm to avoid redundant file uploading [19]. We further check the packet level activities and find that the Dropbox client application (on the data source) is also communicating with the load-balancers to obtain the IP address of Dropbox delivery servers and sending chunk hashes to the delivery servers for comparison. This pre-processing stage is marked as *Stage 1* in the figures, which remarkably elevates the CPU usage without much data transmission.

After pre-processing, we can see that the uploading traffic on the data source starts to increase. In Figure 4, the uploading rate increases to around 1000 KBytes/sec. Meanwhile, the CPU utilization in Figure 1 decreases to around 50%. We mark this as *Stage 2*. It is worth noting that if we compare the 50% CPU utilization in Figure 1 (*Stage 2*) with the 1000 KBytes/sec uploading rate in Figure 4 (*Stage 2*), we can find that the uploading rate cannot cause such a high CPU utilization. An intuitive explanation is that the Dropbox client application is compressing the chunks while uploading. To verify this, we compute the total uploaded bytes and compare it with the original file size. We find that the total uploaded byte is around 255 MBytes which is much smaller compared to the original file size of 500 MBytes. It is easy to see that the file compression efficiently reduces the time cost in uploading.

As we can see from Figure 4, as soon as the data source finishes the uploading, the destination will start downloading (marked as *Stage 3* in the figures) from Dropbox servers. Figure 1 indicates that the CPU utilization at the destination during this stage is much lower compared to the uploading stage at the data source. Since both servers have similar hardware configurations, we believe that this low CPU utilization is because the file decompressing process is much easier than the file compressing[2]. Another noticeable feature is that the downloading will start only when the entire file has been successfully uploaded to the Dropbox servers. The reason is that the file is segmented and compressed on the data source. The Dropbox servers are designed to merge the chunks together and then send the file to storage servers before further delivery. Although this design might not be the most efficient for minimizing synchronization latency, it largely avoids the possible errors during the file uploading and compressing. This distinguishes Dropbox from conventional file hosting systems that are often pipelined [4] [6].

When the downloading stage finishes, the downloading rate at the destination drops to zero. However, Figure 1 shows that the CPU still keeps working for another 60 seconds after the downloading stage, indicating that there is a post-

[2]We also change the data source and destination, and observe that for the same computer, the CPU utilization is lower for downloading.

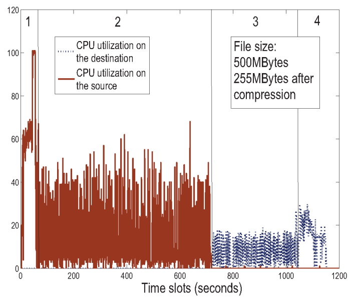

Fig. 1: Client CPU utilization (file size 500 MBytes)

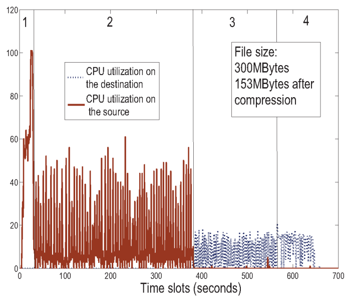

Fig. 2: Client CPU utilization (file size 300 MBytes)

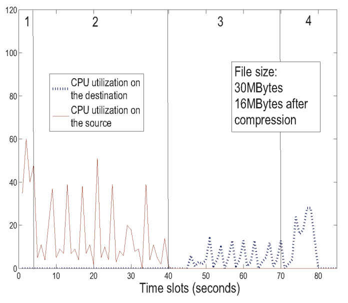

Fig. 3: Client CPU utilization (file size 30 MBytes)

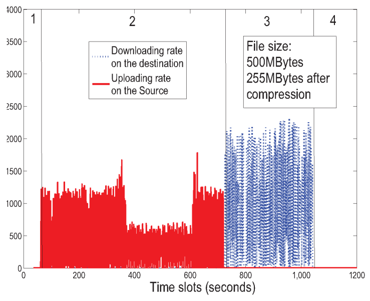

Fig. 4: Client downloading/uploading rate (file size 500 MBytes)

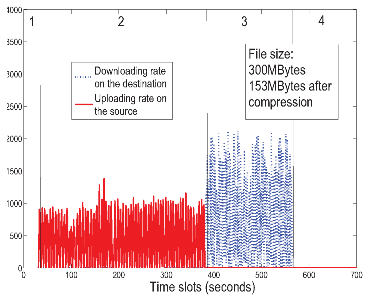

Fig. 5: Client downloading/uploading rate (file size 300 MBytes)

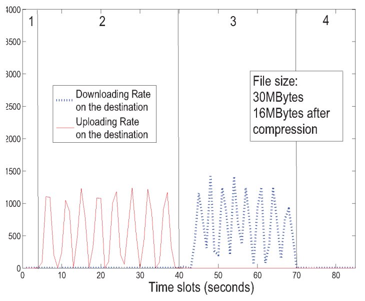

Fig. 6: Client downloading/uploading rate (file size 30 MBytes)

processing stage (marked as *Stage 4*). Since the files are segmented and compressed on the sender side, this CPU load is for decompressing the received chunks and merging them together.

In summary, our measurement shows that the Dropbox file synchronization can be decomposed into 4 cascaded stages: (1) *pre-processing*, (2) *uploading*, (3) *downloading*, and (4) *post-processing*. To generalize this observation and avoid possible bias, we have conducted a series of experiments and present representative results in Figures 2−3 and 5−6. We can see clear stages in all the experiments as marked in these figures. It is also worth noting that in Figure 2 and Figure 6, there is a very small delay (around 3 sec) between the end of uploading and the start of downloading stages. We believe that it is mainly due to the costs of such operations on the Dropbox servers as finding the right VMs for delivery and sending the files to S3. This delay is relatively low (less than 5 seconds in all cases), which can be largely ignored.

While these serial operations look natural, they are apparently less efficient than pipelined operations as in the traditional file storage systems, e.g., [4] [6]. Besides simplicity and better reliability in handling data, the uniqueness of cloud virtualization would be a key reason toward adopting the serial operations. It has been found that, for a virtual machine, the traffic load can largely slow down the computation-intensive tasks (such as compressing/decompressing) and create a severe bottleneck in cloud-based systems [20]. This is because the control and data paths in a virtualized network interface controller (NIC) are much longer than that of a non-virtualized counterpart; e.g., in EC2, the paravirtualized NIC involves 3 CPU interrupts and 3 data copies for receiving one packet, as contrast to 1 interrupt and 1 in a bare-metal NIC. The impact on multi-core CPUs is even higher with extra switches across CPUs to handle the interrupts. To avoid potential interference, the bandwidth-intensive and computation-intensive tasks shall be interleaved without overlap, as Dropbox does. We will examine the effectiveness of this straightforward solution as well as its performance bottlenecks in the following sections.

TABLE I: Synchronization latency with different file size

FILE SIZE	AVG	STD	MAX	MIN
15 MBytes	69.6 sec	3.2	72.0 sec	66.0 sec
30 MBytes	131.3 sec	1.5	133.0 sec	130.0 sec
60 MBytes	256.0 sec	19.2	278.0 sec	242.0 sec
120 MBytes	609.3 sec	41.0	644.0 sec	564.0 sec
240 MBytes	1211.3 sec	64.0	1275.0 sec	1147.0 sec

IV. Performance of File Synchronization

A. *Latency and Scalability*

To understand the performance of Dropbox, we first investigate the synchronization latency between 2 Dropbox users

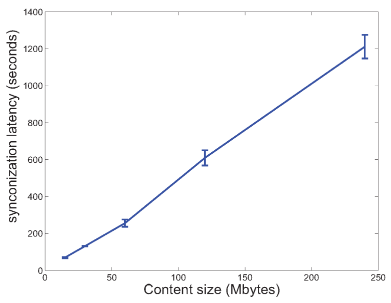

Fig. 7: Synchronization latency of Dropbox

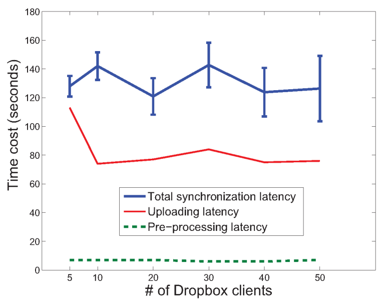

Fig. 8: Time cost on the data source (uploader side)

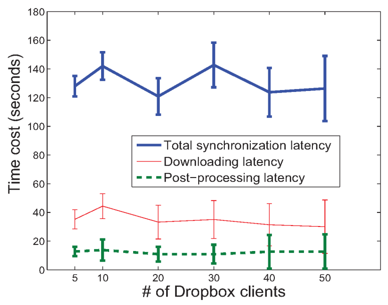

Fig. 9: Time cost on the destination (downloader side)

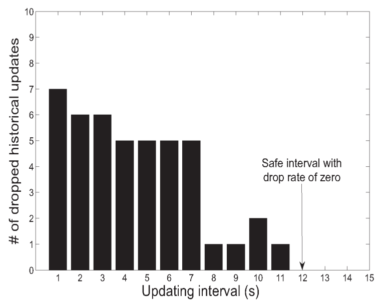

Fig. 10: # of dropped updates with different upload intervals

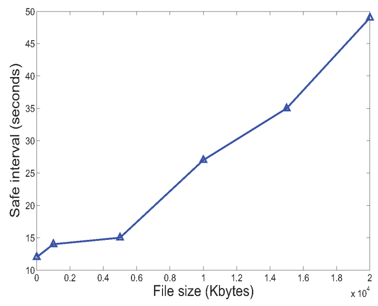

Fig. 11: Safe intervals under different file sizes

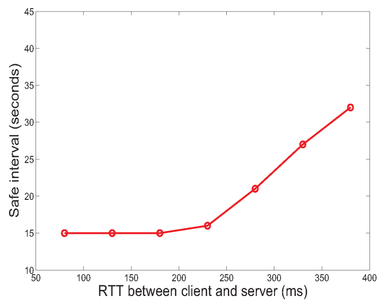

Fig. 12: Safe intervals under different RTTs (file size 5 MBytes)

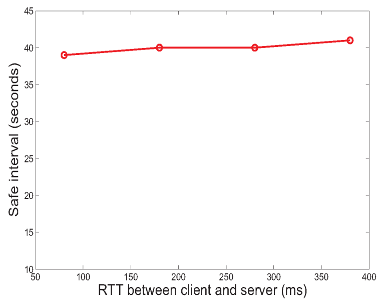

Fig. 13: Safe interval when uploading rate is limited to 100 KBytes/sec (file size 5 MBytes)

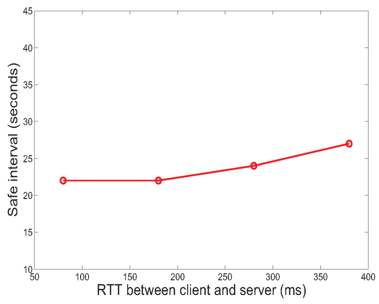

Fig. 14: Safe interval when uploading rate is limited to 200 KBytes/sec (file size 5 MBytes)

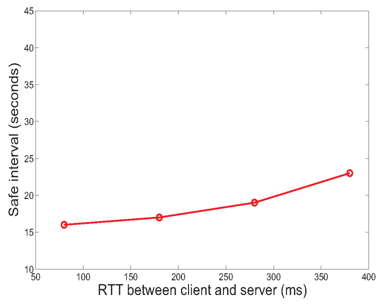

Fig. 15: Safe interval when uploading rate is limited to 300 KBytes/sec (file size 5 MBytes)

with different file sizes. We then increase the number of users to examine its impact.

In the first experiment, the configuration is similar to that in the previous section, but the file size varies. We randomly generate the content of files in each experiment and test the file size of 15 MBytes, 30 MBytes, 60 MBytes, 120 MBytes and 240 MBytes, respectively. We run each experiment 4 times and present the average and standard deviation of synchronization latency in Figure 7. It is easy to see that the synchronization latency increases roughly linearly with the file size. The standard deviation also slightly increases with larger files. This result shows that Dropbox can scale quite well with file size. The detailed statistics can be found in Table I.

To examine the synchronization latency across more Dropbox users, we carry out a Planet-lab based experiment across 51 Dropbox users (one data source and 50 destinations) using a file of 30 MBytes. We use this scale because the advertised capacity of one Dropbox account is currently 40 (based on Dropbox's official documents). Moreover, it is also hard to assume that a user will use one single Dropbox account to synchronize the files across over 50 computers in real-world.

Figures 8 and 9 present the average and standard deviation of time costs of different stages (the four stages that we have mentioned in the last section) at the data source and destinations, respectively. We can see that the average time costs of these four stages are not sensitive to the increasing number Dropbox users. However, the variance of latency across different users increases with the number of users, and thus individual users may face severe performance degradation when the system expands.

We also find that the time cost of file uploading is always more expensive than that of downloading. The pre-processing and the post-processing, on the other hand, are generally quite fast especially compared to data transmission. In particular, the pre-processing of a 500 MBytes file (before compressing) is around 60 seconds, and the post-processing costs only 80 seconds. The pre-/post-processing of smaller files (less than 1 MBytes), will cost even less time (around 10 sec). While this time cost is not significantly high, it can be further optimized via more efficient compressing/decompressing algorithms.

B. Discussion

Our experimental results indicate that the existing Dropbox system scales well with the number of users in terms of average synchronization latency. The variance of the latency however increases, implying that Dropbox is not limiting the transmission rate of different users to achieve equal synchronization latency. Some users can finish the synchronization of a 300 MBytes contents within 10 minutes while others may have to wait for more than 30 minutes. This variance is caused by the different computation capabilities that are responsible for pre-/post-processing, and the different network bandwidth of clients and VMs that are responsible for uploading/downloading. Different from the conventional synchronization techniques that rely on a centralized server, the different VMs in the Dropbox system are highly variant in terms of their network bandwidth which is mainly caused by the interference of computational work load [21]. The significant variance also introduce challenges to the consistence of the synchronization file, which is essential to the version control in user collaboration. It is difficult to ensure fairness for different users given that the VMs are highly distributed. Such unfairness has also been complained by Skydrive and Gdrive users. While this would be acceptable for free services, it can severely hinder the commercialization of such systems for paid users. Our results also suggest that this problem is getting worse when the system scale expands. Providing both fast and fair services, or better yet guaranteed services, to all subscribers remains a challenging task for Dropbox-like systems.

In addition, from Figure 4 and Figure 5, the file uploading from Dropbox client application to the EC2-based Dropbox servers contribute to almost 60% of the synchronization latency. The uploading rates of Dropbox are mostly around 1 to 1.5 MBytes/sec whereas the downloading rates can achieve more than 2 MBytes/sec. We also find that the uploading rates are quite unstable over time (for example, in Figure 16), which is not the case in the downloading stage. Since our experimental platform is dedicated for the measurements, we believe that this is due to the arriving loads of other Dropbox

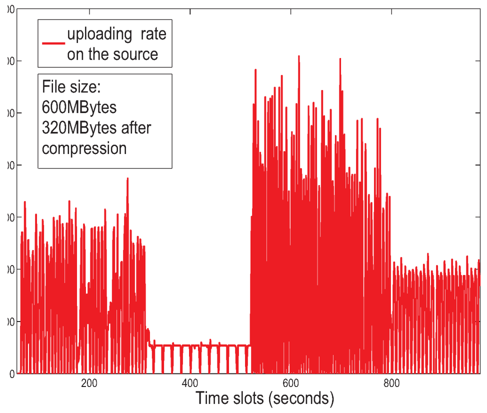

Fig. 16: Uploading rate of the Dropbox client (file size 600 MBytes)

users. To verify this, we have deployed a set of Xen-based[3] local VMs to examine the overhead of both incoming and outgoing traffic. Our experiment shows that the receiving of TCP traffic is more expensive than sending. In particular, the receiving rate of 200 MBytes/sec will cost more than 40% CPU on a virtual machine that has 7.5 GB memory and 2 virtual cores. Meanwhile, the sending rate of 200 MBytes/sec will only cost around 20% CPU on this VM. For example, when more users upload their files to the Dropbox servers, such an increasing receiving traffic will greatly slow down the servers and unavoidably prolong the file processing as well as message forwarding. Such interference potentially creates a severe bottleneck in the system.

V. Impact on User Collaboration

It is encouraging to see that decomposing computation and communication operations does not significantly affect the synchronization latency. Unfortunately, some individual users may face severe performance degradation especially when the system scale grows. For conventional file storage systems, the impact is confined to these individual users. Powered by cloud computing, Dropbox-like systems however also enable such diverse user collaborations as editing and version management, which have been a key factor toward their success. It is thus necessary to examine the impact of heterogeneous users on multi-party collaboration.

A very basic requirement of collaboration is that the system should not discard users' updates without any warning [23]. To this end, Dropbox provides two key functions to handle users' updates: keep conflict versions and record file uploading history [24]. When multiple users are trying to update the same file at the same time, only one copy will be saved as the original update. Other updates, on the other hand, will generate new copies of this file as "conflict version". This is a very intuitive design to avoid the possible loss of users' updates, which is also adopted in such conventional version

[3]Note that Amazon EC2 uses Xen-based virtualization [22]

TABLE II: Fact of view in Figure 11

File size (MB)	0.0001	1	5	10	15	20
Safe interval (s)	12	14	15	27	35	49

TABLE III: Fact of view in Figure 12

Latency (ms)	80	130	180	230	280	330	380
Safe interval (s)	15	15	15	16	21	27	32

TABLE IV: Safe interval under different file sizes with uploading speed of 100 KBytes/sec

File size (MB)	0.0001	1	2	3	4	5	10
Safe interval (s)	12	14	18	23	31	39	80

management applications as Subversion [25]. It is however known that this function also has some problems. For example, the synchronization of conflict files can trigger the generation of more conflict versions and may overwrite some updates from the users. To make sure that the users will not lose any of their updates, Dropbox also provides a file history function that keeps users' historical updates on the servers. According to the official document of Dropbox [26], this function will keep every single change in users' Dropbox folders over the last 30 days for free users and unlimited time for paid users.

A. *Single User Case*

To examine the effectiveness of these functions, we first examine the case when the file updates only come from one Dropbox user. Our objective is to see whether Dropbox can keep all the historical updates for this user even when s/he is slow in terms of the latency. We then extend this extreme case to multiple collaborative users.

In our first experiment, the user updates a small file with the size less than 0.1 KBytes to the Dropbox server. S/he then tries to edit the content of this file for 10 times. We vary the time interval between consecutive updates from 1 to 15 seconds and check the total number of historical files recorded on the Dropbox server. The Round Trip Time (RTT) between this user and the Dropbox server is about 80 ms.

The system should be able to keep all these 10 updates and save them as historical files [26]. However, as we can see from Figure 10, when the updating interval is set to 1 second, Dropbox loses 7 out of the 10 updates from the user. This means that 70% of the user's historical updates are dropped. It is interesting to see that when we slow down the upload interval, the dropping rate will be reduced. When the upload interval is equal to or longer than 12 seconds, Dropbox records all the updates from the user. These results indicate that Dropbox will discard users' updates even when they are quite fast. To avoid this, the user has to wait at least 12 seconds to send new updates. For the ease of discussion, we refer to this interval as *safe interval* in the following sections.

To better understand this problem, we carry out experiments under different file sizes and RTTs. Figure 11 further clarifies

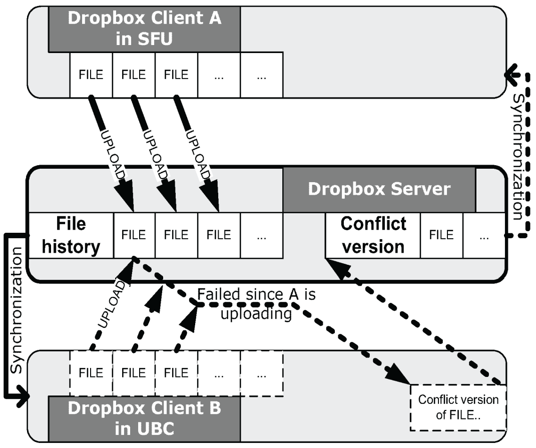

Fig. 17: Experiment across multiple users

the relationship between file size and safe interval. We can again observe that the safe interval will increase with the file size[4].

As we can see from Figure 12, when we increase the RTT between the user and Dropbox servers[5], the safe interval will increase correspondingly. When the server-client latency is equal to 380 ms, the safe interval will be larger than 30 seconds, which is more than twice the safe interval with RTT ranging from 80 to 230 ms. It is also noting that the safe interval quickly elevates when the RTT exceeds 200 ms. This is because the client's uploading rate is largely reduced when the RTT is larger than 200. Considering such a the growth trend, the can easily exceed the auto-saving interval of many applications. It is known that Dropbox has a great number of mobile users with potentially longer latencies [28]. As a result, these users are more likely to lose their updates.

Figures 13, 14, and 15 take a close look into the relationship between the safe and uploading rate. In this experiment, we use *trickle* [29] to limit the maximum uploading rate of user clients. As we can see from Figure 13, a limited uploading rate, say 100 KBytes/sec, will unavoidably lead to large safe intervals around 40 seconds. If we compare Figure 13 with Figure 14, we can see that the client with better uploading capacity, say 200 KBytes/sec, will have smaller safe intervals even with higher client-server RTT. This observation is further confirmed in Figure 15 where the uploading rate is limited to 300 KBytes/sec. We also report the safe interval under different file sizes with the maximum uploading rate of 100 KBytes/sec in Table IV. We can see that the trend of safe interval is more predictable for larger files. When the file size reaches 3 MBytes, the safe interval increases about 8 seconds for an extra MBytes. Recalling that the file is compressed before uploading, with an compression ratio of

[4]Note that we have computed the total amount of the uploading traffic to ensure that all of the 10 updates are successfully/completely uploaded to the Dropbox server. This is to avoid the possible bias due to incomplete uploading.

[5]The RTT is controlled by TC (*Traffic Control Tool*) [27].

0.6, the increased safe interval matches well with the extra uploading traffic plus the communication overhead. We also observe the same phenomenon when the maximum uploading speed is set to 200 and 300 KBytes/sec, respectively.

B. Multiple User Case

Our experiment shows that Dropbox is more likely to discard a user's updates when s/he is slow in terms of the latency. We now examine the multi-user collaboration scenario, in particular, the impact of these high RTT users on the whole session of collaboration.

As shown in Figure 17, our experiment starts with the case of two Dropbox users. One is deployed in our campus (user A) and the other is deployed in UBC (user B). These two Dropbox users are editing the same file and both of them will send 10 updates with constant upload intervals. The RTT of both users (to the Dropbox server) is around 80 ms and the file size is less than 0.1 KBytes. Figure 17 shows an ideal example in the Dropbox system where all the users' updates can be recorded on the Dropbox server, either in the original file history or as conflict versions. In this example, user A's updates are always successfully uploaded (recorded in historical files) and user B's updates are always generating conflict versions. The solid lines show the data flow from user A and the dotted lines indicate the data flow from user B. The Dropbox server will save a total of 20 copies where 10 of them are recorded as conflict versions (from B) and the other 10 as normal updates (from A) in the original file history. If any of these updates are missing, the users' collaboration will be affected.

As shown in Figure 18, we can see that the Dropbox again drops the updates from users during collaborations. If we compare this figure with Figure 10, we can see that the safe interval increases due to the collaboration between the two users. In particular, Figure 10 shows that Dropbox can record all the updates from a single user when his/her upload interval is larger than 12 sec. If there are two users editing this file, 12 sec can however no longer guarantee this. Their updates cloud be dropped unless they slow down the upload interval to more than 17 sec. Figures 19 and 20 present the details of these saved updates, including whose updates are saved as historical updates and whose updates are generating conflict versions. By checking Figure 19 with the upload interval (x-axis) of 6 sec, we can see that 4 updates from the SFU user and 2 updates from the UBC user are saved as normal updates. From Figure 20 (when the x-axis is equal to 6 sec), we can again find that 1 update from the SFU user and 4 updates from the UBC user are recorded as conflict version files. Therefore, the Dropbox system has dropped 5 out of 10 updates from the SFU user and 4 out of 10 updates from the UBC user[6]. Based on our experiments, Dropbox will drop the updates from both users with no determined patterns. This indicates that the Dropbox system is not designed to discard these updates on purpose. This unpredictable dropping behavior will unavoidably bring significant challenges to user collaboration.

Figure 21 further indicates that such a problem will become worse when we increase the RTT for only one user (with fixed file size less than 1 KByte). We can see that the safe interval of both users will increase. When the RTT exceeds 300 ms, both users have to wait over 40 seconds to send their new updates. Otherwise, their updates could be dropped by the system without any warning. We can find that the overall system performance is severely confined by the slowest collaborator. Unfortunately, the series of computation and communication operations (discussed in Section IV and V) increases the variance of latency across different users. This variance however largely reduces the trustworthy of the collaboration across all users.

Figure 22 indicates that the file size will also affect users' safe interval (the RTTs of both users are fixed to 80 ms). We can see that compared with the safe interval with only user, the safe interval is noticeably longer. For example, the safe interval will be around 17 and 57 seconds when the file size is set to 0.1 KBytes and 20 MBytes, respectively, which is 5 and 8 seconds more than that with only one user, respectively.

Figure 23 further shows that the safe interval increases when there are more users in larger-scale collaborations (with RTT fixed to 80 ms and file size less than 0.1 KByte). The matter will become even worse when there are some high latency users in this larger-scale collaboration. Our experiment shows that if 1 out of these 10 users have the latency of 300 ms, the safe interval of all users will easily exceed 1 minutes.

VI. Why Are Updates Dropped?

Randomly dropping updates is a nightmare for both Dropbox operator and its users. A intuitive explanation is that the users are updating too fast, exceeding the safe interval and hence the service capacity that Dropbox can offer. The solution is therefore that the safe interval should be enforced for critical updates/users. This is apparently only a temporary solution, not the ultimate one that the users or we expect.

To unveil the root causes of the dropping, we revisit the two types of communications in Dropbox file synchronization: the communication between users (both uploaders and downloaders) and EC2 servers as well as the communication between EC2 and S3 servers. Since we have already ensured the uploading between users and EC2 servers (by comparing the total amount of uploading traffic with the file size). It is thus reasonable to believe that the problem is due to the communication between EC2 and S3 servers. To better understand this, we carry out some follow-up experiments to see when the updates are more likely to be dropped. Figure 24 shows an example of our explanation. In this figure, the gray boxes denote the Dropbox users and servers; the black boxes denote the file (chunks), and the solid lines show the data flow of file transmission. We also use dotted lines to mark the time slots from t_0 to t_7.

In this example, user A is holding a file and starts to synchronize this file at time t_0. This file is fully uploaded to the

[6]We use these figures to show the existence of such a problem. We have done the experiment multiple times and the results (average values) can be found in our dataset at http://netsg.cs.sfu.ca/dropboxdata.html.

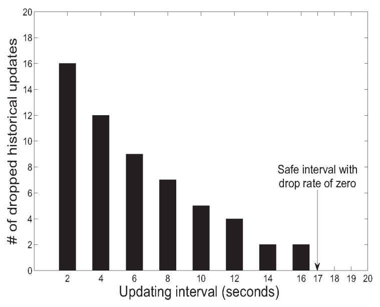

Fig. 18: # of dropped updates with different upload intervals (2 users)

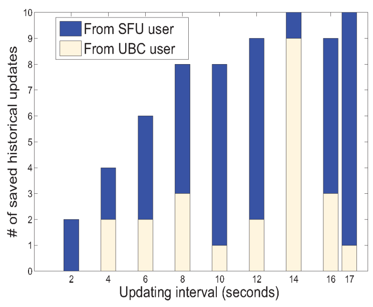

Fig. 19: # of normal updates from different users

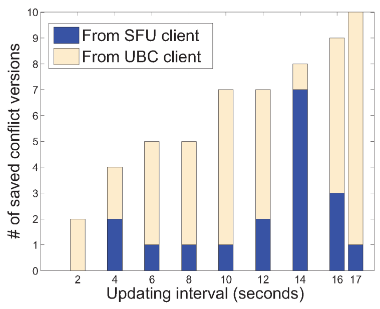

Fig. 20: # of conflict versions from different users

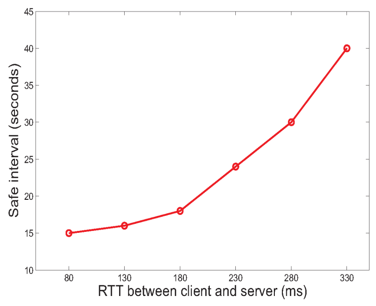

Fig. 21: Safe interval under different RTTs with file size of 5 MBytes (2 users)

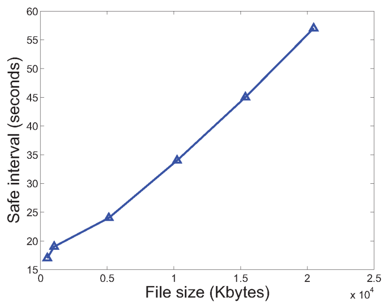

Fig. 22: Safe interval under different file sizes (2 users)

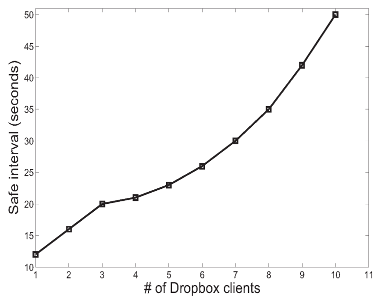

Fig. 23: Safe interval with more users

Dropbox EC2 server at time t_3 and arrive at another user B at time t_7. Note that the protocols between EC2 and S3 servers are unknown. We therefore assume that the communication between EC2 and S3 servers also starts at time t_3. This is an reasonable assumption because Figure 1 and Figure 6 indicate that the gap between uploading and downloading stage is indeed very small.

If another Dropbox user, say user C, is trying to upload the same file between t_0 and t_3 to the EC2 server. This action will trigger the generation of conflict versions since the EC2 server is receiving this file from user A. When these conflict versions (or other normal updates) are further synchronized between time t_3 and t_7, they are very likely to be dropped. This also explains why the safe interval is related to the RTT and file size, as well as related to the total number of Dropbox users.

Some earlier leaked information suggested that Dropbox was planning to move some of their servers out of Amazon and to upgrade to a hybrid service framework [30]. Although this needs further confirmation and our measurement shows that it has not been implemented, at least for now, a hybrid service framework across more cloud platforms could be a better alternative. It potentially makes Dropbox servers closer to the users, which not only speeds up file synchronization but also reduces the drop rate of users' updates. To validate this, we have run experiment on Planet-lab nodes. We selected 20 nodes and put them into two groups. The first group consists of users who are closer to the Dropbox servers (with the maximum RTT being less than 90 ms and average hop counts within 10). The second group consists of users who are relatively far from the Dropbox servers (with the maximum RTT being over 300 ms and average hop over 13). We test the collaboration in these two groups separately on Dropbox. Each user generates 10 updates with upload interval of 50 sec. As shown in Figure 25, we can see that the drop rate can be noticeably reduced when the Dropbox servers are closer to the users. Previously, over 30% high RTT users suffer from a drop rate over 50%; the drop rate decreases to less than 20% if the users are closer to the servers. This observation suggests that a single-data center-based solution might not be the best for the cloud synchronization/collaboration systems; server placement and selection play critical roles for further optimization.

VII. Conclusion

This paper investigated the protocols as well as the performance of Dropbox-like cloud file synchronization systems. We particularly focused on the synchronization/collaboration in such a system, and examined new issues and challenges due to the cloud deployment.

Our study represents an initial attempt toward understanding such a new generation of service. We expect that our findings help with optimizing these systems as well as with migrating

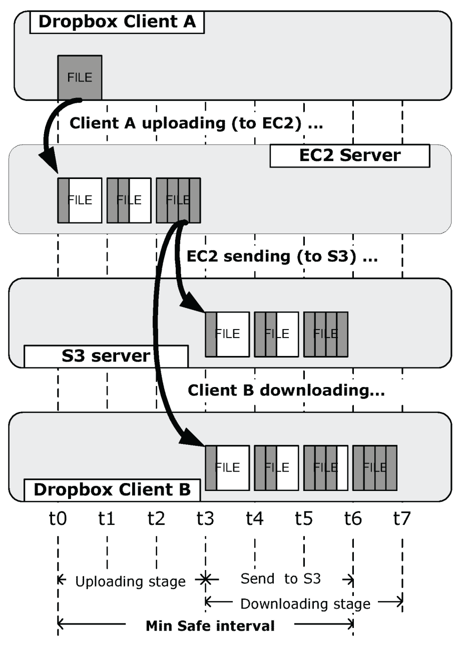

Fig. 24: Analysis of safe interval

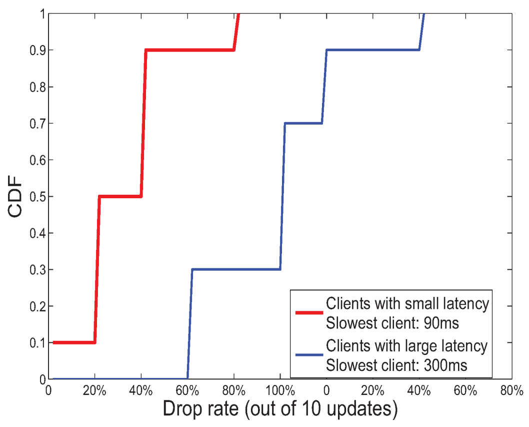

Fig. 25: Drop rate of different users

more Internet services to cloud platforms. There are many possible future directions to explore. We are particularly interested in efficient and scalable collaboration among the users with decentralized cloud deployment, as well as examining other similar applications to further generalize our findings.

References

[1] Dropbox. [Online]. Available: https://www.dropbox.com/
[2] Gdrive. [Online]. Available: https://drive.google.com/
[3] Skydrive. [Online]. Available: http://windows.microsoft.com/skydrive/home
[4] E. K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim, "On the Scalability of Data Synchronization Protocols for PDAs and Mobile Devices," *IEEE Network, 16(4) 22-28, 2002.*
[5] P. M. Chen, E. K. Lee, G. A. Gibson, R. H. Katz, and D. A. Patterson, "RAID: High-performance, Reliable Secondary Storage," *ACM Computing Surveys, 26(2), 1994.*
[6] F. Liu, Y. Sun, B. Li, and X. zhang, "FS2You: Peer-Assisted Semi-Persistent Online Hosting at a Large Scale," *IEEE Transactions on Parallel and Distributed Systems, 21(10) 1442-1457, 2010.*
[7] J. S. Ward, "A Performance Comparison of Clouds: Amazon EC2 and Ubuntu Enterprise Cloud," *Proc. SICSA DemoFEST, 2009.*
[8] A. Li, X. Yang, S. Kandula, and M. Zhang, "CloudCmp: Comparing Public Cloud Providers," *Proc. USENIX Workshop on Hot Topics in Cloud Computing (HotCloud), 2010.*
[9] T. Benson, A. Akella, A. Shaikh, and S. Sahu, "CloudNaaS: A Cloud Networking Platform for Enterprise Applications," in *Proc. ACM SOCC, 2011.*
[10] Y. Wu, C. Wu, B. Li, X. Qiu, and F. Lau, "CloudMedia: When Cloud On Demand Meets Video On Demand," in *Proc. IEEE ICDCS, 2011.*
[11] S. Kannan, A. Gavrilovska, and K. Schwan, "Cloud4Home – Enhancing Data Services with Home Clouds," in *Proc. IEEE ICDCS, 2011.*
[12] Y. Seung, T. Lam, L. E. Li, and T. Woo, "Seamless Scaling of Enterprise Applications into The Cloud," in *Proc. IEEE INFOCOM, 2011.*
[13] U. Sharma, P. Shenoy, S. Sahu, and A. Shaikh, "Kingfisher: Cost-aware Elasticity in the Cloud," in *Proc. IEEE ICDCS, 2011.*
[14] B. Raghavan, K. Vishwanath, S. Ramabhadran, K. Yocum, and A. C. Snoeren, "Cloud Control with Distributed Rate Limiting," in *Proc. ACM SIGCOMM, 2007.*
[15] P. Singh, M. Lee, S. Kumar, and R. R. Kompella, "Enabling Flow-level Latency Measurements across Routers in Data Centers," in *Proc. USENIX Workshop on Hot Topics in Management of Internet(HotICE) , 2011.*
[16] Sugarsync. [Online]. Available: https://www.sugarsync.com/
[17] I. Drago, M. Mellia, M. M. Munafo, A. Sperotto, R. Sadre, and A. Pras, "Inside Dropbox: Understanding Personal Cloud Storage Services," in *Proc. ACM conference on Internet measurement conference (IMC), 2012.*
[18] Z. Li, C. Jin, T. Xu, C. Wilson, Y. Liu, L. Cheng, Y. Liu, Y. Dai, and Z. Zhang, "Towards Network-level Efficiency for Cloud Storage Services," in *Proc. ACM SIGCOMM Internet Measurement Conference (IMC), 2014.*
[19] Dropbox Wiki. [Online]. Available: http://en.wikipedia.org/wiki/Dropbox-service/
[20] R. Shea and J. Liu, "Understanding the Impact of Denial of Service Attacks on Virtual Machines," in *Proc. IEEE/ACM International Workshop on Quality of Service (IWQoS), 2012.*
[21] H. Wang, R. Shea, F. Wang, and J. Liu, "On the Impact of Virtualization on Dropbox-like Cloud File Storage/Synchronization Services," in *Proc. IEEE/ACM International Workshop on Quality of Service (IWQoS), 2012.*
[22] Xen-based Amazon EC2. [Online]. Available: http://en.wikipedia.org/wiki/Amazon_Elastic_Compute_Cloud/
[23] How to Use Dropbox as a Killer Collaborative Work Tool. [Online]. Available: http://lifehacker.com/5792938/how-to-use-dropbox-as-a-killer-collaborative-work-tool/
[24] How do I recover old versions of files? [Online]. Available: https://www.dropbox.com/help/11/en/
[25] SVN. [Online]. Available: http://subversion.apache.org/
[26] Dropbox File History. [Online]. Available: https://www.dropbox.com/help/11/en/
[27] Networking and Traffic Control On Linux. [Online]. Available: http://tcng.sourceforge.net/
[28] Dropbox Anywhere. [Online]. Available: https://www.dropbox.com/anywhere/
[29] trickle. [Online]. Available: http://monkey.org/~marius/pages/?page=trickle/
[30] Dropbox Move Out of EC2? [Online]. Available: http://www.quora.com/Amazon-EC2/What-sites-moved-out-of-EC2-and-where-did-they-go/